Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №4
«Функциональные возможности языка Rust. Бот для парса МосБиржи»

Выполнил:
студент группы ИУ5-32Б
Фролов Илья
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Подпись и дата:

Москва, 2024 г.

Задание:

Разработайте простого бота для Telegram. Бот должен использовать http запросы на внешний сервер для получения данных.

Текст программы:

utils/json_processing.rs

DzProject > src > utils > json_processing.rs > { } json > get_private_data > s

```rust
pub mod json {
    fn create_valid_json_value(value: String) -> String {
        value[1..value.len() - 1].to_string()
    }

    pub fn get_private_data() -> serde_json::Value {
        let got_data: Result<String, std::io::Error> =
            std::fs::read_to_string(path: "protected/data.json");
        let s: String = match got_data {
            Ok(s: String) => s,
            Err(_) => panic!("Can't read file"),
        };
        serde_json::from_str(&s).expect(msg: "Can't parse json")
    }

    pub fn get_token(secure_data: &serde_json::Value) -> String {
        create_valid_json_value(secure_data["privateData"]["token"].to_string())
    }
}

```

crates/bot.rs

DzProject > src > crates > bot.rs > start_bot > bot

```rust
use crate::crates::requests::*;
use teloxide::prelude::*;

#[tokio::main]
pub async fn start_bot(token: String) -> Result<(), Box<dyn std::error::Error>> {
    log::info!("Starting bot...");

    let bot: Bot = Bot::new(token);

    teloxide::repl(bot, |bot: Bot, msg: Message| async move {
        let moex_data: MOEXResponse = get_moex_data().await.unwrap();
        if moex_data.securities.data.is_empty() {
            bot.send_message(chat_id: msg.chat.id, text: "Нет данных о ценах валют").await?;
        } else {
            let text: String = moex_data.securities.data.iter().map(|security: &Vec<String>| security.join(sep: ", ")).collect::<Vec<_>>().join(sep: "\n");
            bot.send_message(chat_id: msg.chat.id, text).await?;
        }
        Ok(())
    })
.await;

    Ok(())
}
```

crates/handler.rs

DzProject > src > crates > handler.rs > { } handler

```rust
pub mod handler {
    /* TODO: сохранение пользователей в БД и проверка на необходимость отправки
    use mongodb::{
        bson::{doc, Document},
        Client, Collection,
    };*/
}
```

Crates/requests.rs

DzProject > src > crates > requests.rs > get_moex_data

```rust
use serde::{Deserialize, Serialize};
use anyhow::Result;

#[derive(Serialize, Deserialize, Debug)]
3 implementations
pub struct MOEXResponse {
    pub securities: MOEXSecurities,
}

#[derive(Serialize, Deserialize, Debug)]
3 implementations
pub struct MOEXSecurities {
    pub columns: Vec<String>,
    pub data: Vec<Vec<String>>,
}

pub async fn get_moex_data() -> Result<MOEXResponse> {
    let client: Client = reqwest::Client::new();
    let resp: MOEXResponse = client Client
    .get(url: "https://iss.moex.com/iss/engines/currency/markets/selt/boards/united/securities.json") RequestBuilder
    .send() impl Future<Output = Result<…, …>>
    .await? Response
    .json::<MOEXResponse>() impl Future<Output = Result<…, …>>
    .await?;
    Ok(resp)
}
```

Sensitive-данные:

DzProject > protected > {..} data.json > { } privateData

```json
{
    "privateData": {
        "mongoUrl": "mongodb://127.0.0.1:27017",
        "mongoDbName": "",
        "token": "935459293:AAHI7XXWVeYehEDR_DCUo8uT4zDOoBX0jCs"
    }
}
```

---

main.rs

DzProject > src > main.rs

```rust
mod utils;
use crate::utils::json_processing::json::*;

mod crates;
use crate::crates::bot::*;

▶ Run | Debug
fn main() {
    pretty_env_logger::init();

    let secure_data: serde_json::Value = get_private_data();
    let token: String = get_token(&secure_data);

    let _error: Result<(), Box<dyn Error>> = start_bot(token);
    log::info!("Stoping bot...");
}

```

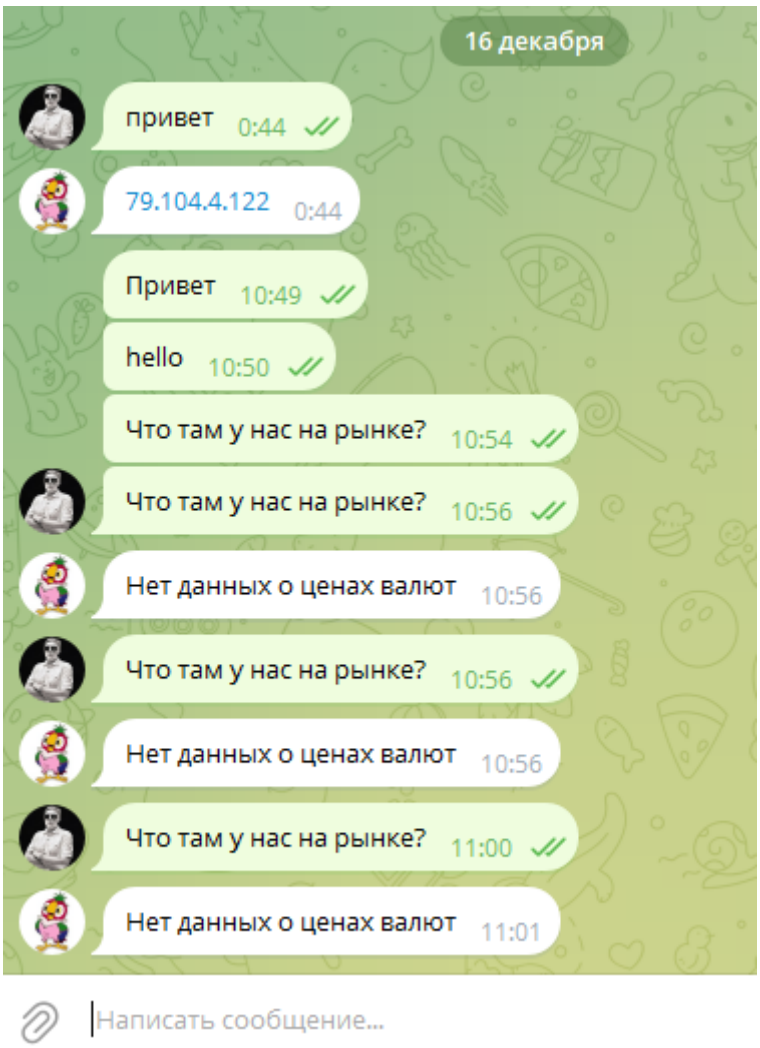
Зависимости:

Cargo.toml × | main.rs | requests.rs | bot.rs | handler.rs | mod.rs ...\crates | json_processing.rs | mod.rs ...\utils | .gitignore | data.json | README.md

DzProject > Cargo.toml

```toml
[package]
name = "DzProject"
version = "0.1.0"
edition = "2021"

[dependencies]
# http requests
reqwest = { version = "0.11", features = ["json"] }
# json
serde_json = {version="1.0.99"}

# mongodb
serde = "1.0.188"
futures = "0.3.28"

# telegram
teloxide = { version = "0.13", features = ["macros"] }
log = "0.4"
pretty_env_logger = "0.5"

# union dependencies
tokio = { version =  "1.8", features = ["full", "rt-multi-thread", "macros"] }

# error handler
anyhow = "1.0.68"

[dependencies.mongodb]
version = "3.1.0"
```

Результаты исполнения:


16 декабря
привет 0:44
79.104.4.122 0:44
Привет 10:49
hello 10:50
Что там у нас на рынке? 10:54
Что там у нас на рынке? 10:56
Нет данных о ценах валют 10:56
Что там у нас на рынке? 10:56
Нет данных о ценах валют 10:56
Что там у нас на рынке? 11:00
Нет данных о ценах валют 11:01
Написать сообщение...